《C/C++学习指南》

第29.1讲：函数模板

作者：邵发　　QQ群：417024631
官网：http://afanihao.cn

背景

在一个int型数组中，查找最大的数。。。

```
int findmax (int arr[],  int len)
{
      int  val = arr[0];
      for(int i=1; i<len ; i++)
      {
            if(arr[i] > val) val = arr[i];
      }
      return  val;
}
```

背景

在一个double型数组中，查找最大的数。。。
在一个float型数组中，查找最大的数。。。
在一个Object[]数组中，查找最大的元素。。。

算法都是一样的：遍例数组，找出最大值。只是元素的类型不一样而已。

如果每一种类型都重载一个findmax函数，是否有点太笨拙了呢？？

函数模板

模板，template：定义一个模子，自动适应各个类型。

语法：

```
template <typename T>
T findmax (T arr[],  int len)
{
      T val = arr[0];
      ....
}
```

① 算法相同
② 元素类型不同，用T代替

函数模板的使用

使用时，用<>来具体指定typename的类型

```
int main()
{
	int  arr[ 4] = { 1, 42,  87, 100 };
	int  result = findmax <int> (arr,  4);
	return 0;
}
```

也就是说，在使用时函数名为 **findmax<int>** ，表示把int类型代入模板。

（联想一下“情书生成器”“西太平洋大学文凭生成器” ）

函数模板

由于C/C++里的，空白是不影响最终编译结果的，所以以下两个风格结果是一样的。

第一种风格：(推荐风格)

```
template <typename T>
T findmax (T arr[],  int len)
{
}
```

第二种风格：

```
template <typename T>  T findmax (T arr[],  int len)
{
}
```

小结

（1）使用函数模板的好处：相同的算法就不用重复的写多遍了。简化了代码。

（2）函数模板用于实现通用的算法 generic algrithm。有的教程上称为泛型算法。“通用算法”

（3）通常，我们更多的是调用人家写好的函数模板，很少自己去写一个模板。